Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1999 Proceedings

Americas Conference on Information Systems (AMCIS)

December 1999

Systems Architectonics

Galal Galal
University College London, UK

Ray Paul
Brunel University

Systems Architectonics

Galal Hassan Galal
E-mail: Galal@acm.org
Software Systems Engineering Group
Department of Computer Science
University College London, Gower Street, London WC1E 6BT,
United Kingdom.

Ray J Paul
E-mail: Ray.Paul@Brunel.ac.uk
Department of Information Systems and Computing
Brunel University
Uxbridge, Middlesex UB8 3PH
United Kingdom

Abstract

In Architecture, a particular model of "learning buildings" was proposed that comprises six constructional layers that change at different rates. The more these layers can evolve without requiring changes to other layers, the more adaptable the building is. By analogy, organising the constructional components in an information system into layers that share similar likelihood of change may lead to systems that are easier to adapt in the face of change. If this view is accepted, then the act of analysing and designing information systems should have as a central aim the identification of layers or categories of domain elements that have similar rates of change within any one layer.

Introduction

In the world we live in, change is inevitable. Environments change. Governments and their legislations change. Technology changes. Organisations change. Individuals change. Any living organism must be able to adapt to changes in its surroundings; poor adaptability may mean extinction. Modern information systems that serve the changing needs of a variety of clients also need to change. The ease with which a system can be changed is what has been traditionally referred to as adaptability. The software and information systems communities have for some time been addressing the issue of designing adaptable systems. Structure has been one important prong of the attack. Reduction of complexity and coupling, and increasing cohesion of structural elements, collectively referred to as functional independence, have been such structurally-orientated directions. But the relevance of these tends to be limited to the design and construction phases. At which time it is probably too late to achieve a greater degree of domain-specific system adaptability. We argue that such thinking should start much earlier in the IS development process: at the requirements analysis stage.

One important source of the need for adapting software and information systems is the constant change in requirements: new functional and quality characteristics need to be accommodated by the system. The question for the information systems community is whether efforts to enhance the adaptability characteristic of information systems can be started at the earlier stages of systems analysis and requirements specification. How can the traditional systems analysis activity, with its constituent analysis and modelling tasks, be oriented to support the definition of adaptable system? We look at the discipline of Architecture, and within it the concept of "adaptable building", which features a layered view of building construction, in an attempt to approach this question.

How do buildings adapt?

Stewart Brand in a book entitled "How buildings learn, what happens after they're built" (Brand, 1994) discusses how buildings adapt, learn and evolve over time. Brand identifies six layers in a building that change at different rates. From the slowest to the fastest changing, these are: Site, Structure, Skin, Services, Space Plan, and Stuff (meaning things such as furniture, decorations, appliances etc.). This view is fundamentally normative, i.e. it is based on what actually happens to buildings after they are built when adapted by their users. However, the response of various buildings to adaptation attempts varies. Buildings that allow such adaptations gracefully, to use Brand's terminology, are the most successful and are the ones that please their users. Buildings that resist adaptation and change, by virtue of

their architectures, wither away or are adapted at great cost. It is important for the architect to understand this normative fact, and design his or her buildings to accommodate the most frequent changes, for example, in space plans. Over time, building users tend to desire changes in the space plans, due for example to changes in family circumstances or working conditions. The design imperative that Brand draws is that an adaptive building "has to allow slippage between the differently-paced [layers] of Site, Structure, Skin, Services, Space Plan, and Stuff". The cost of changing some of these layers is less than others. Some layers are so costly to change, that changing them amounts to complete re-building, for example, the structure layer. When re-modelling a building, the most constraining factors, after the site that is, is the structure. The Structure layer of a building constrains, in a variety of ways, what adaptations can be made to a building for the rest of its life. The corollary to this is that Structure, as a base layers, gives rise to the adaptive properties of the building. Structure implicitly generates a possible set of future adaptations.

This layered view represents of course a very general model, and viewing buildings in this way adds little to what most people know about buildings. From the point of view of software and information systems, no particular architectural model can be generalised to all systems types. The moral remains that, to allow technical artefacts to adapt over a period of time, it is useful to de-couple generic layers of components as far as possible, so as to allow them to slip past each others at different rates. In this way, design trade-offs would favour connections among elements belonging to the same layer over connections between components that belong to different layers, even though a certain amount of the latter is inevitable. The data that inform these decisions, however, come from analysis work. For a specific problem, targeted analysis needs to identify, and separate, layers that share change rates, with reference to the substantive domain of interest. That is of course, if a layered architecture is seen to be the most suitable to the various requirements of the system being engineered.

Architecting systems

Our notion of architecting systems revolves around the premise that it is the task of the information systems engineer (or architect) to explore the architectonic nature of the domain of interest, during the analysis stage. Namely, the engineer needs to identify the generic areas of interest that comprise the domain and their potential dynamics over time, especially in relation to one another. A domain of interest should be regarded as a universe that has concepts that evolve and mutate at different speeds and in different ways. As a consequence, analysing a system and specifying its requirements should not be about comprehensively predicting all the changes that might befall a system, rather, it should be about establishing what model components are core or fundamental to the mission of the system, and which are more transient and are thus more likely to change. Core components would be the last area of the system that needs to change. Grouping these together means that changes to other parts of the system are less disruptive, and hence easier and cheaper to implement.

For example, in the mobile telecommunications domain, new services and price plans emerge all the time, the information system components relating to these interests should collectively, as a layer, be minimally coupled to the rest of the components in the system. Another layer that also changes, albeit at a slower pace, is the components that manage how mobile telephone calls are established and terminated (we see a very similar example in the ISO OSI telecommunications reference model). Again, these components should be in a separate layer that is minimally coupled to other layers in the system. Meanwhile, the set of components that manage basic customer data are likely to be the last to require change, so they are potentially the slowest evolving set of components. As such, we have a set of domain-specific layers, teased apart by their likelihood of changing independently from each other. In this way, analysing the domain to explore how its concepts are differentiated by their rates of change informs later design in a way that enhances the adaptability of the information system.

However, even the result of that exploration need not be single-valued or conclusive. Rather, alternative requirements-led categorisations (or architectures) of system components should be developed, each corresponding to a generic, future scenario. It is therefore vital to build some dynamic, temporal dimension into the analysis. Looking into possible, alternative, futures for the system should inform the analyst and the stakeholders about the areas where flexibility is needed most. One tool that can be useful in this task is that of scenario planning (Shwartz, 1991, quoted in Brand, 1994 in a discussion of the use of scenario planning in building design). Building strategic, long term scenarios with the systems' stakeholders enlightens the parties involved about a range of possible eventualities of a system. Which in turn brings to their attention which categories of the system components need to be kept light-weight, because they are most likely to change under certain circumstances. These scenarios should be represented in a way that is both accessible to the parties involved, as well as easily modifiable. In this way, an all important link is made between information systems engineering, and systems engineering in its wider sense.

Our view can be illustrated with reference to a particular well-known software engineering methodology, which incidentally has a strong object-oriented flavour. This is the Jackson's Systems Development, or JSD

(Jackson, 1983). In JSD, functional requirements are deliberately omitted from the initial analysis stages. The early stages of analysis aim to establish and model the entities that constitute the fundamental "subject matter" of the system. Much attention is given to modelling such entities as basic system processes, which are later connected to the components that aim at achieving specific functional requirements. In Jackson's words, the initial model "implies" a set of possible functions that can be added later to the model without disturbing too much of its components unnecessarily, thus making it more robust in the face of change. As such, certain use potential emerges from the basic architecture. This "emergent" system property is what needs to be studied further, perhaps by looking at existing examples.

What next?

We could then posit two core issue in information systems engineering. The first concerns the methodological tools that allow us to identify, in a domain specific way, a number of categories of systems components that are differentiated by their rates of evolution and change. The number of layers should not be too large, otherwise the very purpose of identifying them will be defeated. The second issue concerns again the methodological tools that enable us to anticipate with some reliability the range of adaptations that a particular layering gracefully allows. This is the analytical or evaluative dimension of software and systems methodologies that sadly remains under-developed in our field; the current practice being more oriented towards the more prescriptive views of methodology (see Galal & Paul (1998) for our argument on this aspect of methodology).The analytical/ evaluative aspect entails the problem of studying the type of emergent systems properties, and allowing the "implications" for various systems architectures to be detected from a model. This latter problem is particularly difficult in the light of the relative absence of studies of the vernacular, or the existing, forms of information systems and their adaptive character. So, let this be a call for conducting a kind of "precedent studies" that architects carry out on types of buildings they are commissioned to design. Information systems engineer also should be concerned with studying generic types of systems architectures and what adaptations they have allowed, or prevented, during their lifetimes.

We must say here that we are not advocating a layered view of system or software architectures as a universal solution for all types of systems. There are types of systems, such as these that require a high degree of concurrency, or security, that may not benefit from a layered architecture in the same way as, say, information retrieval systems. What we are advocating is investigating the "architectonics" of a domain during analysis and requirements specification activities so as to inform later design and construction phases.

Conclusions

Some components of an information system are more fundamental to its mission than others. These components are likely to change at a slower pace than the remaining ones, and should therefore be isolated in separate layers during the design phase so as to shield them from changes in other places. Establishing which components have what change characteristics should be included as part of what an information systems engineer does when he or she analyses the domain under consideration. We have looked in this paper at the discipline of Architecture, which has also been concerned with issues of change. We have argued in our analysis that information systems engineering should explicitly adopt as one of its major activities the investigation of "domain architectonic" issues, if it is to inform the design activity in a way that helps the design of information systems that are more adaptable and robust in the face of the inevitable need for later change.

References

Brand, S. (1994). How buildings learn: What happens after they're built (2nd ed.). London: Phoenix Illustrated.

Broadbent, G. (1990). Emerging Concepts in Urban Space Design. London: E & FN Spon.

Galal, G. H. (1998). Software architecting: from requirements to building blocks within an architectural style. In I. Borne, M. Prieto, F. Brito e Abreu, & W. De Meuter (Eds.), 12th European Conference on Object-Oriented Programming: ECOOP'98, July 20-24, 1998. Workshop W2: Techniques, Tools and Formalisms for capturing and assessing Architectural Quality in Object-Oriented Software, Brussels, Belgium.

Galal, G. H., & Paul, R. (1998). From the Prescriptive to the Analytic: A New Direction for Information Systems Engineering Methodology. In D. Naumann & J. DeGross (Eds.), Fourth Americas Conference on Information Systems, (pp. 829-831). Baltimore, Maryland, USA: Association for Information Systems.

Jackson, M. (1983). Systems Development. London: Prentice-Hall International.

Kazman, R., Abowd, G., Bass, L., & Clements, P. (1996). Scenario-Based Analysis of Software Architecture. IEEE Software(November 1996), 47-55.

Shwartz, P. (1991). The Art of the Long view. New York: Doubleday.